

WHAT IS CLAIMED IS:

- 1 1. A computer-implemented method of scheduling threads
2 for a plurality of SMT processors, said method
3 comprising:
4 determining that a first thread in a first run queue
5 is a poor performing thread, wherein the first run
6 queue corresponds to a first SMT processor;
7 in response to the determination:
8 writing a first identifier corresponding to the
9 first thread to a second run queue, wherein the
10 second run queue corresponds to a second SMT
11 processor; and
12 removing the first identifier from the first run
13 queue.
- 1 2. The method as described in claim 1 further comprising:
2 determining that a second thread in the second run
3 queue is another poor performing thread;
4 in response to the determination regarding the second
5 thread:
6 writing a second identifier corresponding to the
7 second thread to the first run queue; and
8 removing the second identifier from the second
9 run queue.
- 1 3. The method as described in claim 1, wherein the
2 determination further comprises:

3 executing a plurality of threads listed in the first
4 run queue, including the first thread, on the first
5 SMT processor, the executing further including:

6 retrieving a number of cycles value for each
7 thread indicating the number of cycles that
8 occurred while each thread was executing;

9 retrieving a number of instructions value for
10 each thread indicating the number of instructions
11 that were executed while each thread was
12 executing;

13 dividing each number of cycles value by its
14 corresponding number of instructions value, the
15 dividing resulting in the CPI value; and

16 recording the CPI value for each thread in the
17 first queue; and

18 identifying the first thread as having a CPI value
19 worse than a plurality of the other threads listed in
20 the first run queue.

1 4. The method as described in claim 3 further comprising:

2 determining whether each thread was previously moved
3 from one of a plurality of SMT processors' run queues
4 to the first SMT processor's run queue.

1 5. The method as described in claim 4 further comprising:

2 determining that the CPI of the first thread has
3 degraded since being moved to the first SMT
4 processor's run queue.

1 6. The method as described in claim 3 further comprising:
2 recording the first thread's identifier, the first
3 thread's CPI, and a timestamp to a previously swapped
4 data structure.

1 7. The method as described in claim 6 wherein the
2 timestamp corresponds to a time selected from the
3 group consisting of a time that the first thread's CPI
4 was calculated, and a time that the first thread's
5 identifier was moved from the first run queue to the
6 second run queue.

1 8. The method as described in claim 3 further comprising:
2 averaging the CPI value for each of the plurality of
3 threads with one or more previous CPI values
4 previously calculated for each of the threads, wherein
5 the recorded CPI value includes the average CPI value
6 for each thread.

1 9. The method as described in claim 1 further comprising:
2 skipping one or more worse performing threads in
3 comparison to the first thread in response to
4 determining that each of the worse performing threads
5 has previously been moved to the first SMT processor's
6 run queue and each has improved in performance since
7 being moved; and
8 identifying the first thread after skipping the worse
9 performing threads.

1 10. The method as described in claim 1 further comprising:

2 sensing that the first thread is about to complete;
3 computing a CPI value for the first thread upon its
4 completion;
5 determining that the CPI value is worse than a
6 threshold value;
7 in response to the determination, checking whether the
8 first thread was previously moved to the first SMT
9 processor from a previous SMT processor selected from
10 a plurality of SMT processors that includes the first
11 SMT processor and the second SMT processor;
12 in response to determining that the first thread was
13 previously moved, determining that the first thread's
14 CPI is worse on the first SMT processor than it was on
15 the previous SMT processor; and
16 wherein the writing and the removing steps are
17 performed in response to determining that the first
18 thread's CPI is worse on the first SMT processor than
19 on the previous SMT processor.

1 11. An information handling system comprising:

2 a plurality of SMT processors;
3 a memory accessible by the processors;
4 a plurality of software threads and a plurality of run
5 queues stored in the memory, wherein each of the run
6 queues corresponds to one of the SMT processors;
7 a thread scheduling tool for scheduling threads among
8 the plurality of SMT processors, the thread scheduling
9 tool comprising software code effective to:

determine that a first thread from the plurality of software threads that is in a first run queue selected from the plurality of run queues is a poor performing thread, wherein the first run queue corresponds to a first SMT processor selected from the plurality of SMT processors;

in response to the determination, the software code is effective to:

write a first identifier corresponding to the first thread to a second run queue, wherein the second run queue corresponds to a second SMT processor selected from the plurality of SMT processors; and

remove the first identifier from the first run queue.

12. The information handling system as described in claim 11 wherein the software code is further effective to:

determine that a second thread in the second run queue is another poor performing thread;

in response to the determination regarding the second thread:

write a second identifier corresponding to the second thread to the first run queue; and

remove the second identifier from the second run queue.

1 13. The information handling system as described in claim
2 11, wherein the determination further comprises
3 software code effective to:

4 execute a plurality of threads listed in the first run
5 queue, including the first thread, on the first SMT
6 processor, the execution comprising software code
7 effective to:

8 retrieve a number of cycles value for each thread
9 indicating the number of cycles that occurred
10 while each thread was executing;

11 retrieve a number of instructions value for each
12 thread indicating the number of instructions that
13 were executed while each thread was executing;

14 divide each number of cycles value by its
15 corresponding number of instructions value, the
16 division resulting in the CPI value; and

17 record the CPI value for each thread in the first
18 queue; and

19 identify the first thread as having a CPI value worse
20 than a plurality of the other threads listed in the
21 first run queue.

1 14. The information handling system as described in claim
2 13 wherein the software code is further effective to:

3 determine whether each thread was previously moved
4 from one of a plurality of SMT processors' run queues
5 to the first SMT processor's run queue.

1 15. The information handling system as described in claim
2 14 wherein the software code is further effective to:
3 determine that the CPI of the first thread has
4 degraded since being moved to the first SMT
5 processor's run queue.

1 16. The information handling system as described in claim
2 13 wherein the software code is further effective to:
3 record the first thread's identifier, the first
4 thread's CPI, and a timestamp to a previously swapped
5 data structure.

1 17. The information handling system as described in claim
2 16 wherein the timestamp corresponds to a time
3 selected from the group consisting of a time that the
4 first thread's CPI was calculated, and a time that the
5 first thread's identifier was moved from the first run
6 queue to the second run queue.

1 18. The information handling system as described in claim
2 13 wherein the software code is further effective to:
3 average the CPI value for each of the plurality of
4 threads with one or more previous CPI values
5 previously calculated for each of the threads, wherein
6 the recorded CPI value includes the average CPI value
7 for each thread.

1 19. The information handling system as described in claim
2 10 wherein the software code is further effective to:

3 skip one or more worse performing threads in
4 comparison to the first thread in response to
5 determining that each of the worse performing threads
6 has previously been moved to the first SMT processor's
7 run queue and each has improved in performance since
8 being moved; and
9 identify the first thread after skipping the worse
10 performing threads.

1 20. The information handling system as described in claim
2 11 wherein the software code is further effective to:
3 sense that the first thread is about to complete;
4 compute a CPI value for the first thread upon its
5 completion;
6 determine that the CPI value is worse than a threshold
7 value;
8 in response to the determination, check whether the
9 first thread was previously moved to the first SMT
10 processor from a previous SMT processor selected from
11 the plurality of SMT processors that includes the
12 first SMT processor and the second SMT processor;
13 in response to determining that the first thread was
14 previously moved, determine that the first thread's
15 CPI is worse on the first SMT processor than it was on
16 the previous SMT processor, wherein the software code
17 to write and remove are performed in response to
18 determining that the first thread's CPI is worse on
19 the first SMT processor than on the previous SMT
20 processor.

1 21. A computer program product stored on a computer
2 operable media for scheduling threads for a plurality
3 of SMT processors, said computer program product
4 comprising:

5 means for determining that a first thread in a first
6 run queue is a poor performing thread, wherein the
7 first run queue corresponds to a first SMT processor;

8 in response to the determination:

9 means for writing a first identifier
10 corresponding to the first thread to a second run
11 queue, wherein the second run queue corresponds
12 to a second SMT processor; and

13 means for removing the first identifier from the
14 first run queue.

1 22. The computer program product as described in claim 21
2 further comprising:

3 means for determining that a second thread in the
4 second run queue is another poor performing thread;

5 in response to the determination regarding the second
6 thread:

7 means for writing a second identifier
8 corresponding to the second thread to the first
9 run queue; and

10 means for removing the second identifier from the
11 second run queue.

1 23. The computer program product as described in claim 21,
2 wherein the means for determining further comprises:

3 means for executing a plurality of threads listed in
4 the first run queue, including the first thread, on
5 the first SMT processor, the means for executing
6 further including:

7 means for retrieving a number of cycles value for
8 each thread indicating the number of cycles that
9 occurred while each thread was executing;

10 means for retrieving a number of instructions
11 value for each thread indicating the number of
12 instructions that were executed while each thread
13 was executing;

14 means for dividing each number of cycles value by
15 its corresponding number of instructions value,
16 the dividing resulting in the CPI value; and

17 means for recording the CPI value for each thread
18 in the first queue, and

19 means for identifying the first thread as having a CPI
20 value worse than a plurality of the other threads
21 listed in the first run queue.

1 24. The computer program product as described in claim 23
2 further comprising:

3 means for determining whether each thread was
4 previously moved from one of a plurality of SMT
5 processors' run queues to the first SMT processor's
6 run queue.

- 1 25. The computer program product as described in claim 24
2 further comprising:
3 means for determining that the CPI of the first thread
4 has degraded since being moved to the first SMT
5 processor's run queue.
- 1 26. The computer program product as described in claim 23
2 further comprising:
3 means for recording the first thread's identifier, the
4 first thread's CPI, and a timestamp to a previously
5 swapped data structure.
- 1 27. The computer program product as described in claim 26
2 wherein the timestamp corresponds to a time selected
3 from the group consisting of a time that the first
4 thread's CPI was calculated, and a time that the first
5 thread's identifier was moved from the first run queue
6 to the second run queue.
- 1 28. The computer program product as described in claim 23
2 further comprising:
3 means for averaging the CPI value for each of the
4 plurality of threads with one or more previous CPI
5 values previously calculated for each of the threads,
6 wherein the recorded CPI value includes the average
7 CPI value for each thread.
- 1 29. The computer program product as described in claim 21
2 further comprising:

means for skipping one or more worse performing threads in comparison to the first thread in response to determining that each of the worse performing threads has previously been moved to the first SMT processor's run queue and each has improved in performance since being moved; and

means for identifying the first thread after skipping the worse performing threads.

30. The computer program product as described in claim 21 further comprising:

means for sensing that the first thread is about to complete;

means for computing a CPI value for the first thread upon its completion;

means for determining that the CPI value is worse than a threshold value;

in response to the determination that the CPI value is worse, means for checking whether the first thread was previously moved to the first SMT processor from a previous SMT processor selected from a plurality of SMT processors that includes the first SMT processor and the second SMT processor;

in response to determining that the first thread was previously moved, means for determining that the first thread's CPI is worse on the first SMT processor than it was on the previous SMT processor; and

wherein the means for writing and the means for removing are performed in response to determining that

21 the first thread's CPI is worse on the first SMT
22 processor than on the previous SMT processor.